First Normal Form (1NF) • A relation will be 1NF if each attribute of a table has atomic (single) value. • It states that an attribute of a table cannot hold multiple values. It must hold only singlevalued attribute. • 1NF disallows the multi-valued attribute, composite attribute, and their combinations**. Second Normal Form (2NF)** • In the 2NF, relational must be in 1NF. • All the non-prime attributes should be fully functional dependent on candidate key. (No partial dependencies) If we follow second normal form, then every non-prime attribute should be fully functionally dependent on prime key attribute. That is, if X → A holds, then there should not be any proper subset Y of X, for which Y → A also holds true.**Third Normal Form (3NF)** • A relation is in 3NF if it is in 2NF and no non key attribute is transitively dependent on the primary key. • 3NF is used to reduce the data duplication. It is also used to achieve the data integrity. • If there is no transitive dependency for non-prime attributes, then the relation must be in 3NF. **Fourth Normal Form (4NF)** • A relation will be in 4NF if it is in BCNF and has no multi-valued dependency. • For a dependency A → B, if for a single value of A, multiple values of B exists, then the relation will be a multi-valued dependency. **Boyce-Codd Normal Form (BCNF)** is an extension to 3NF, and is also known as 3.5 Normal Form. It is stricter than 3NF. Rules for BCNF A table is in BCNF if • Every functional dependency X → Y, X is the super key of the table. • For BCNF, the table should be in 3NF, and for every FD, LHS is super key.

EMP_ID	EMP_NAME	EMP_PHONE	EMP_STATE

EMP_ID	EMP_NAME	EMP_PHONE	EMP_STATE

Before 2NF
Table : Customer

CUST_ID	STORE_ID	LOCATION
1	1	Delhi
1	3	Mumbai
2	1	Delhi
3	2	Bangalore
4	3	Mumbai

Candidate Key: {Cust_Id, Store_Id}

Non prime attribute: Location

After 2NF
Table : Customer_Detail

CUST_ID	STORE_ID
1	1
1	3
2	1
3	2
4	3

Primary Key :

{Cust_Id, Store_Id}

After 2NF
Table : Store_Detail

STORE_ID	LOCATION
1	Delhi
2	Bangalore
3	Mumbai

Primary Key :

{Store_Id}

Vendor

ID	Name	Account_No	Bank_Code_No	Bank

Startsituation

Result after normalisation

Vendor

ID	Name	Account_No	Bank_Code_No

Bank

Bank_Code_No	Bank

Before 4NF
Student

SID	COURSE	HOBBY

After 4NF
Student_Course

SID	COURSE

After 4NF
Student_Hobby

SID	HOBBY

Before BCNF
Teacher
PK : {SID, Subject}

SID	Subject	Professor
101	Java	Chandrakanth
101	C++	Bhuvan
102	Java	Pallavi

After BCNF
Student_Detail
PK : {SID, PID}

SID	PID
101	1
101	2
102	3

After BCNF
Professor_Detail
PK : {PID}

PID	Professor	Subject
1	Chandrakanth	Java
2	Bhuvan	C++
3	Pallavi	Java

Explain informal design guidelines for relation schemas. **(8marks)**

The four informal measures of quality for relation schema

i. Semantics of relations attributes

Specifies how to interpret the attributes values stored in a tuple of the relation. In other words, how the attribute value in a tuple relate to one another.

Guideline 1: Design a relation schema so that it is easy to explain its meaning. Do not combine attributes from multiple entity types and relationship types into a single relation.

ii. Reducing redundant values in tuples. Save storage space and avoid update

Guideline 2: Design the base relation schemas so that no insertion, deletion, or modification anomalies occur. Reducing the null values in tuples. e.g., if 10% of employees have offices, it is better to have a separate relation, EMP_OFFICE, rather than an attribute OFFICE_NUMBER in EMPLOYEE.

iii. Reducing the null values in tuples

NULLs can have multiple interpretations, such as the following:

- The attribute does not apply to this tuple. For example, Visa_status may not apply to U.S. students.
- The attribute value for this tuple is *unknown*. For example, the Date_of_birth may be unknown for an employee.
- The value is known but absent; that is, it has not been recorded yet. For example, the Home_Phone_Number for an employee may exist, but may not be available and recorded yet.

Guideline 3: Avoid placing attributes in a base relation whose values are mostly null.

iv. Generation of Spurious Tuples: Spurious Tuples are the tuples that are not in the original relation but generated by natural join of decomposed subrelations.

Guideline 4: Design relation schemas so that they can be naturally JOINed on primary keys or foreign keys in a way that guarantees no spurious tuples are generated.

Discuss insertion , deletion and modification anomalies . Why are they considered bad? Illustrate with example **(08 Marks)**

Insertion Anomalies :-

It can be differentiated in to two types illustrated by the following examples based on the EMP-DEPT relation.

- To insert a new employee tuple into EMP-DEPT, we must include either the attribute values for the department that the employee works for or Nulls.
- For example , to insert a new tuple for an employee who works in department number 5, we must enter all the attribute values of department 5 correctly so that they are consistent with the corresponding values for department 5 in other tuples in EMP-DEPT.
- It is difficult to inserts new department that has in employee as yet in the EMP-DEPT relation. The only way to do this is to place Null values in the attributes for employee . This violates the entity integrity for EMP-DEPT because SSN is its primary key.

Deletion Anomalies :-

The problem of deletion Anomalies is related to the second insertion anomaly situation . If we delete from EMP-DEPT an employee tuple that happens to represent the last employee working for a particular department, the information concerning that department is lost from the database.

Modification Anomalies :-

In EMP-DEPT , if we change the value of one of the attributes of a particular department ,the manager of department 5, we must update the tuples of all employees who works in that department, otherwise the database will be inconsistent. If we fail to update tuples, the same department will be shown to have two different values for manger on different employee tuples, which would be wrong.